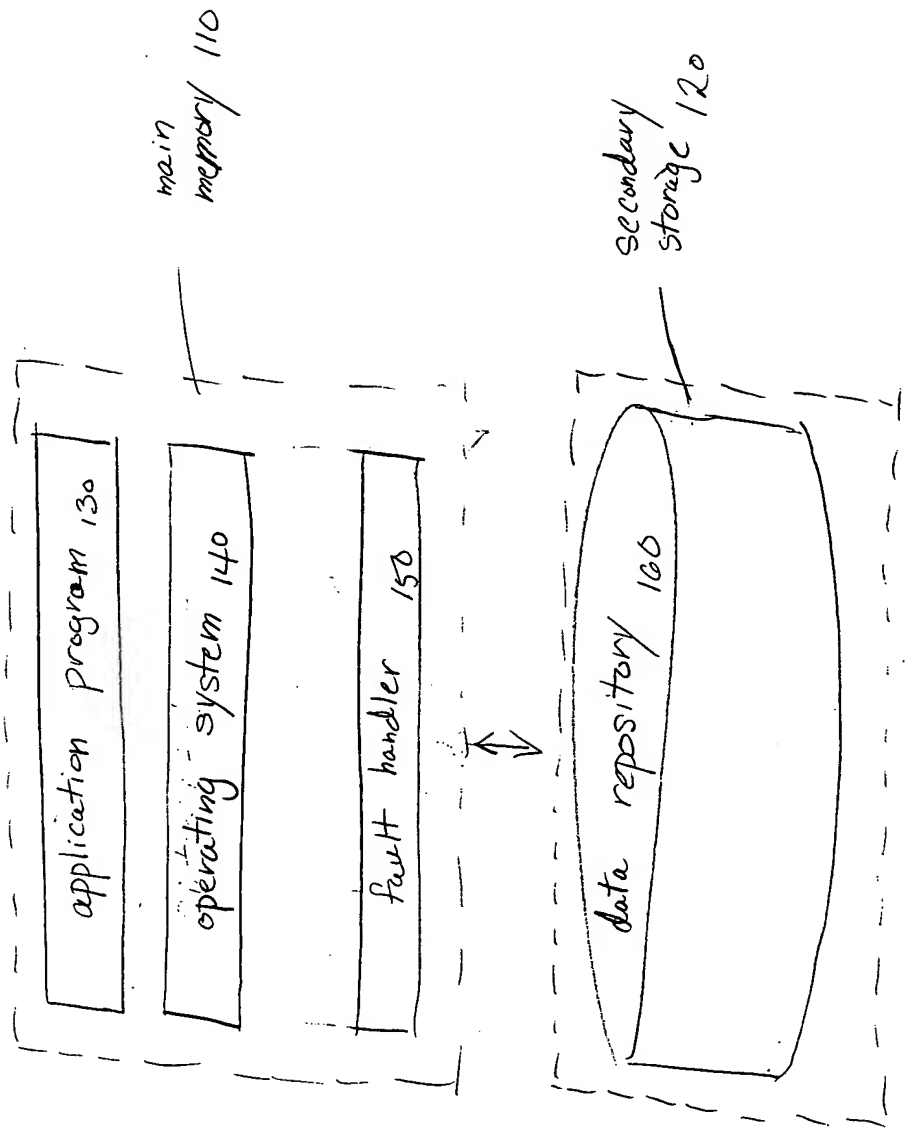
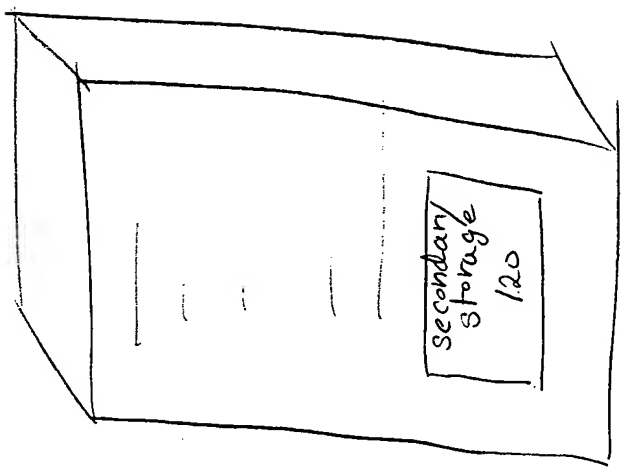
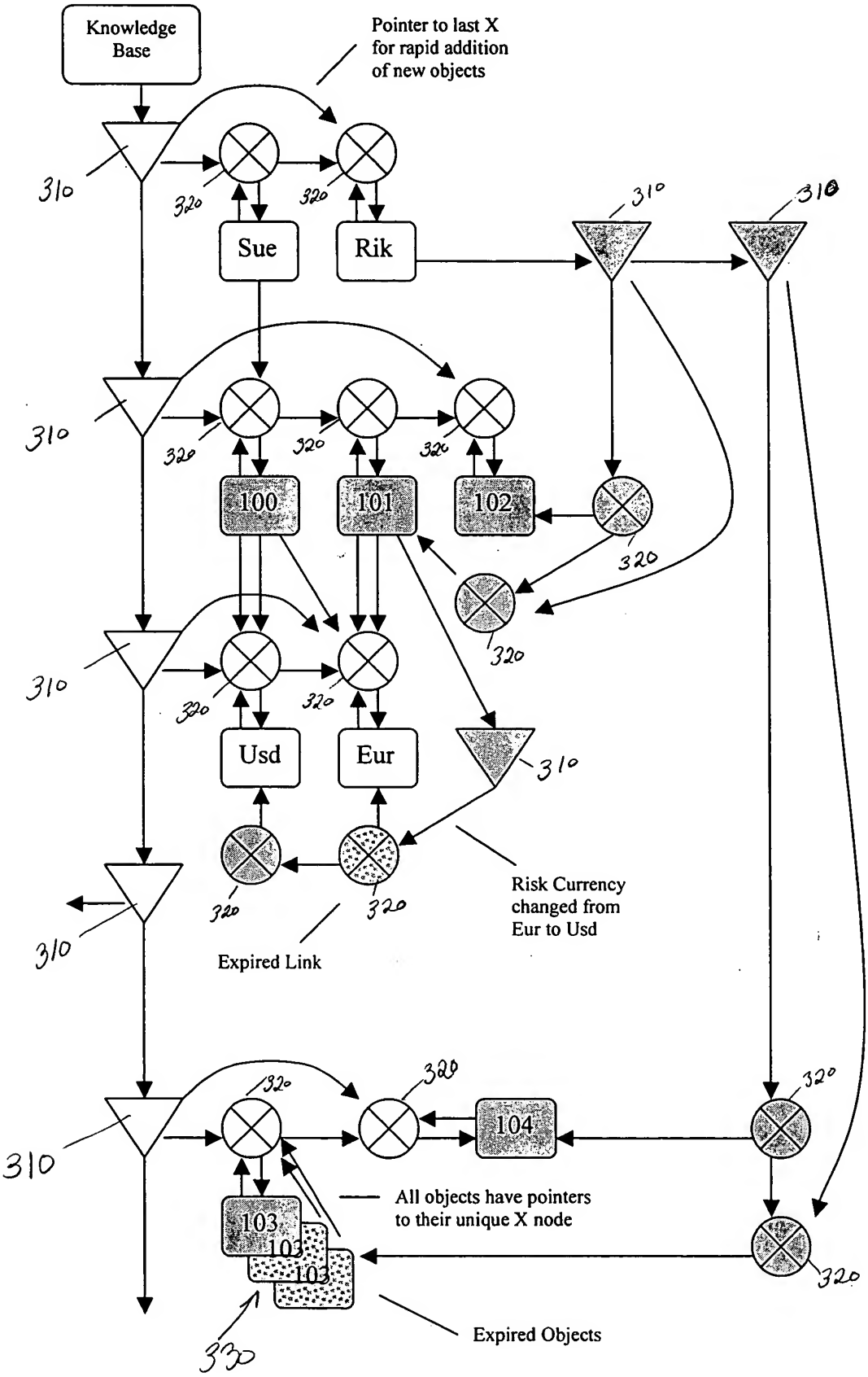

main memory 110
secondary storage 120
application program 130
operating system 140
fault handler 150
data repository 160
100

FIG. 1


secondary storage
120
application and database server
210

200

FIG. 2


Knowledge Base
Pointer to last X for rapid addition of new objects
310
320
Sue
Rik
100
101
102
Usd
Eur
Risk Currency changed from Eur to Usd
Expired Link
104
All objects have pointers to their unique X node
103
Expired Objects
330

300

FIG. 3

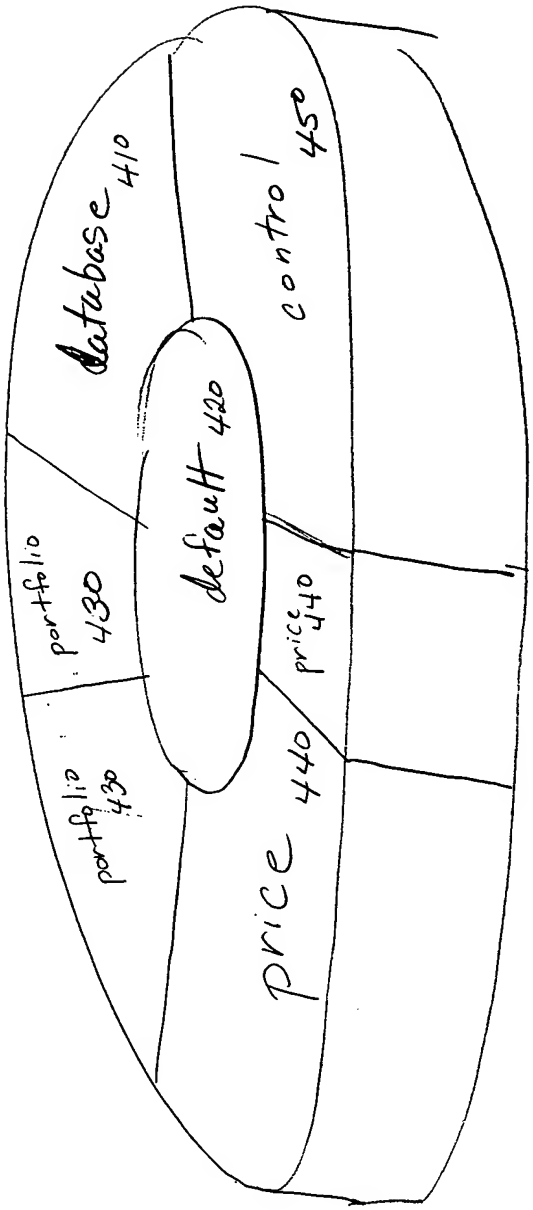
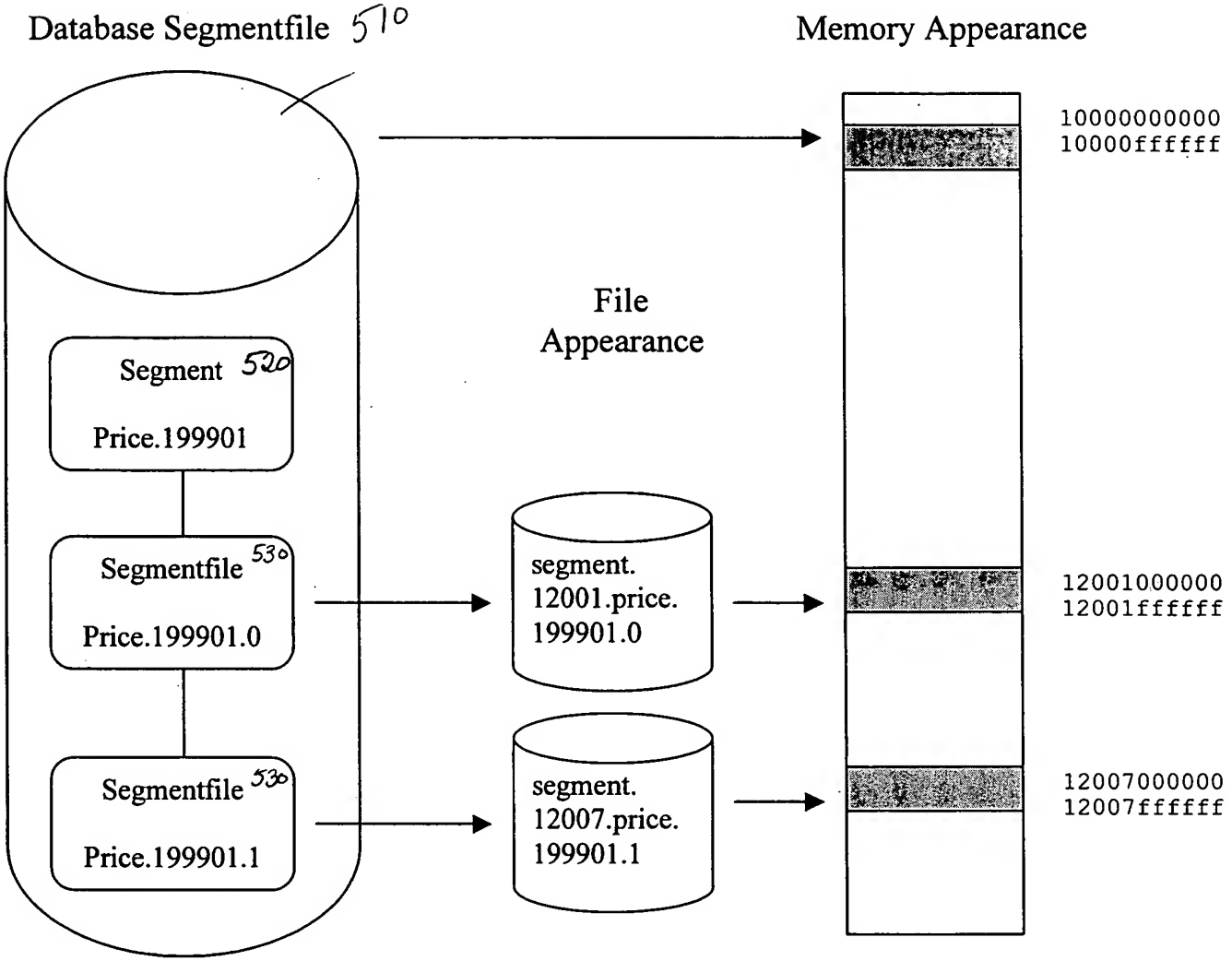

database 410
control 450
default 420
portfolio 430
price 440
portfolio 430
price 440

FIG. 4

400


Database Segmentfile 510
Memory Appearance
10000000000
10000ffffff
File
Appearance
Segment 520
Price.199901
Segmentfile 530
Price.199901.0
segment.
12001.price.
199901.0
12001000000
12001ffffff
Segmentfile 530
Price.199901.1
segment.
12007.price.
199901.1
12007000000
12007ffffff

500

FIG. 5

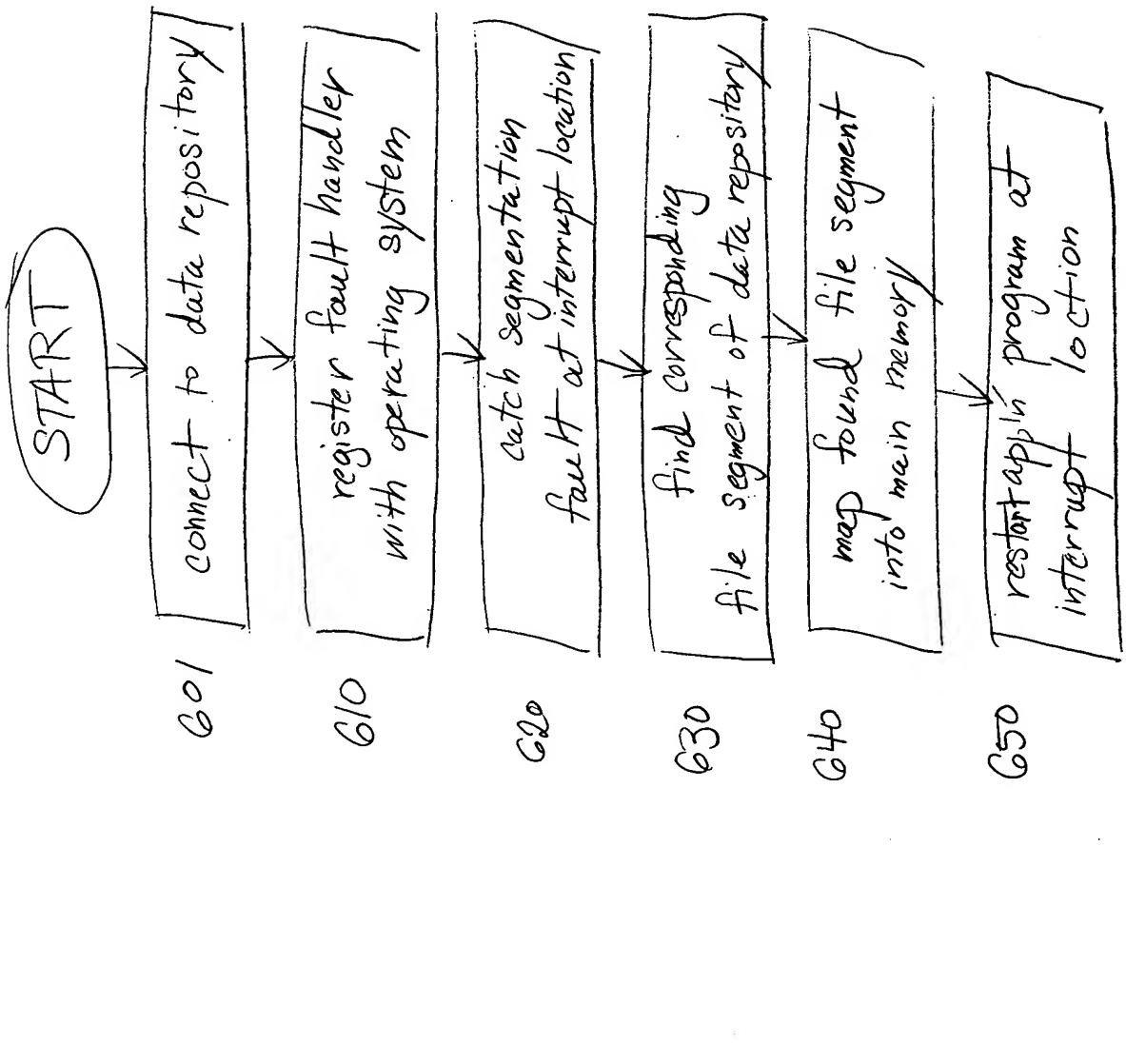
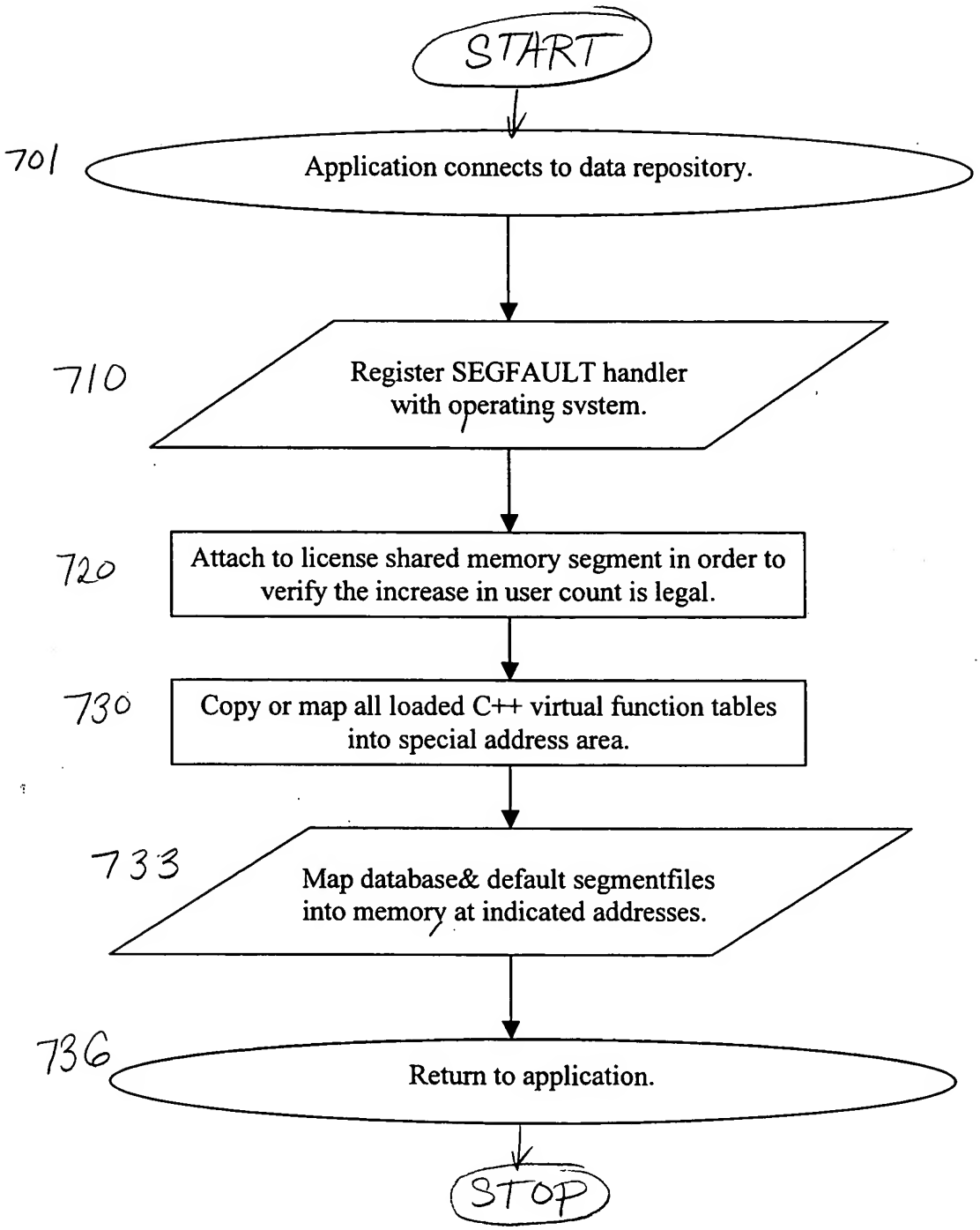

START
connect to data repository
601
register fault handler with operating system
610
catch segmentation fault at interrupt location
620
find corresponding file segment of data repository
630
map found file segment into main memory
640
restart application program at interrupt location
650
600

FIG. 6


START
701
Application connects to data repository.
710
Register SEGFAULT handler with operating system.
720
Attach to license shared memory segment in order to verify the increase in user count is legal.
730
Copy or map all loaded C++ virtual function tables into special address area.
733
Map database& default segmentfiles into memory at indicated addresses.
736
Return to application.
STOP

700

FIG. 7A

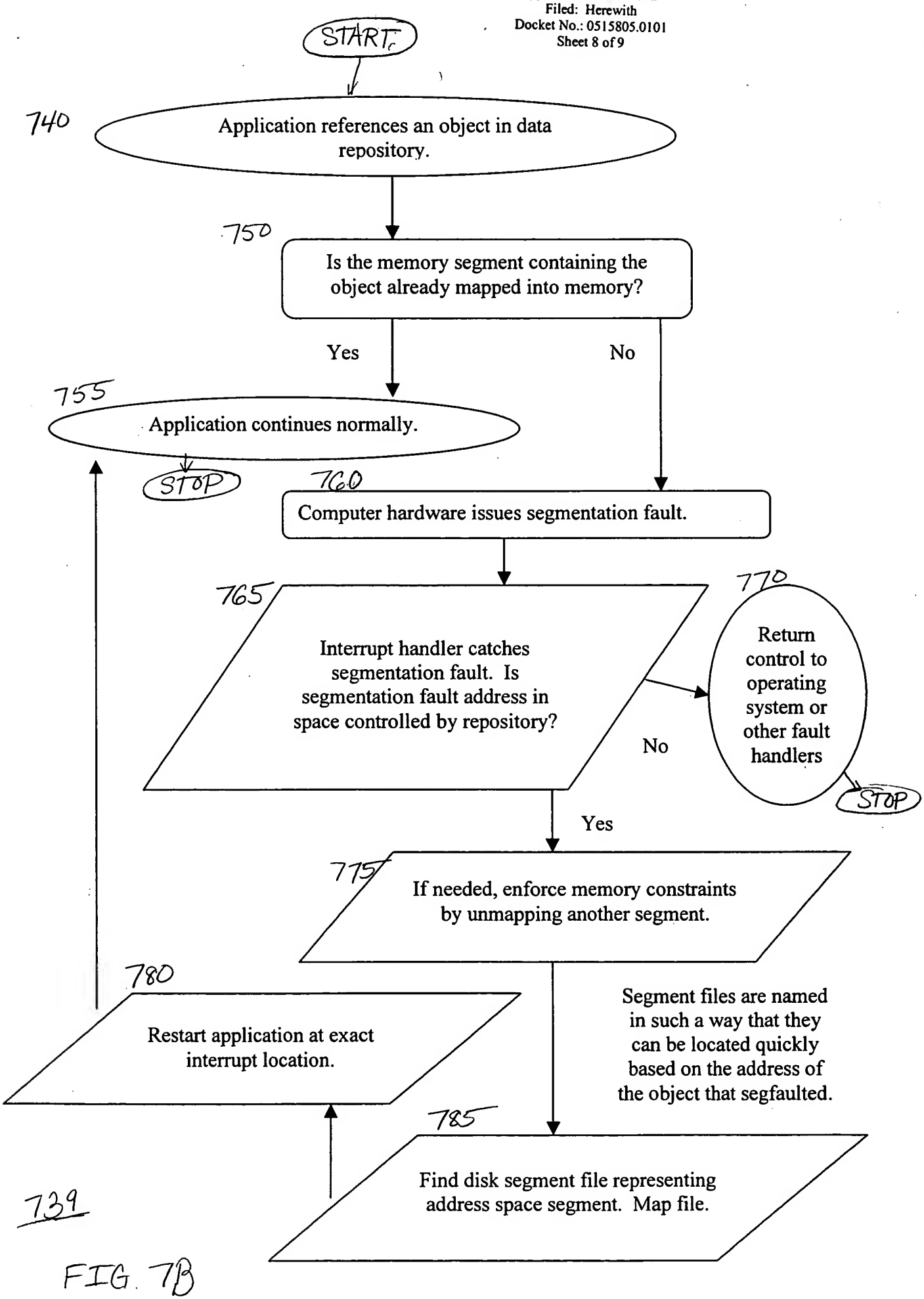
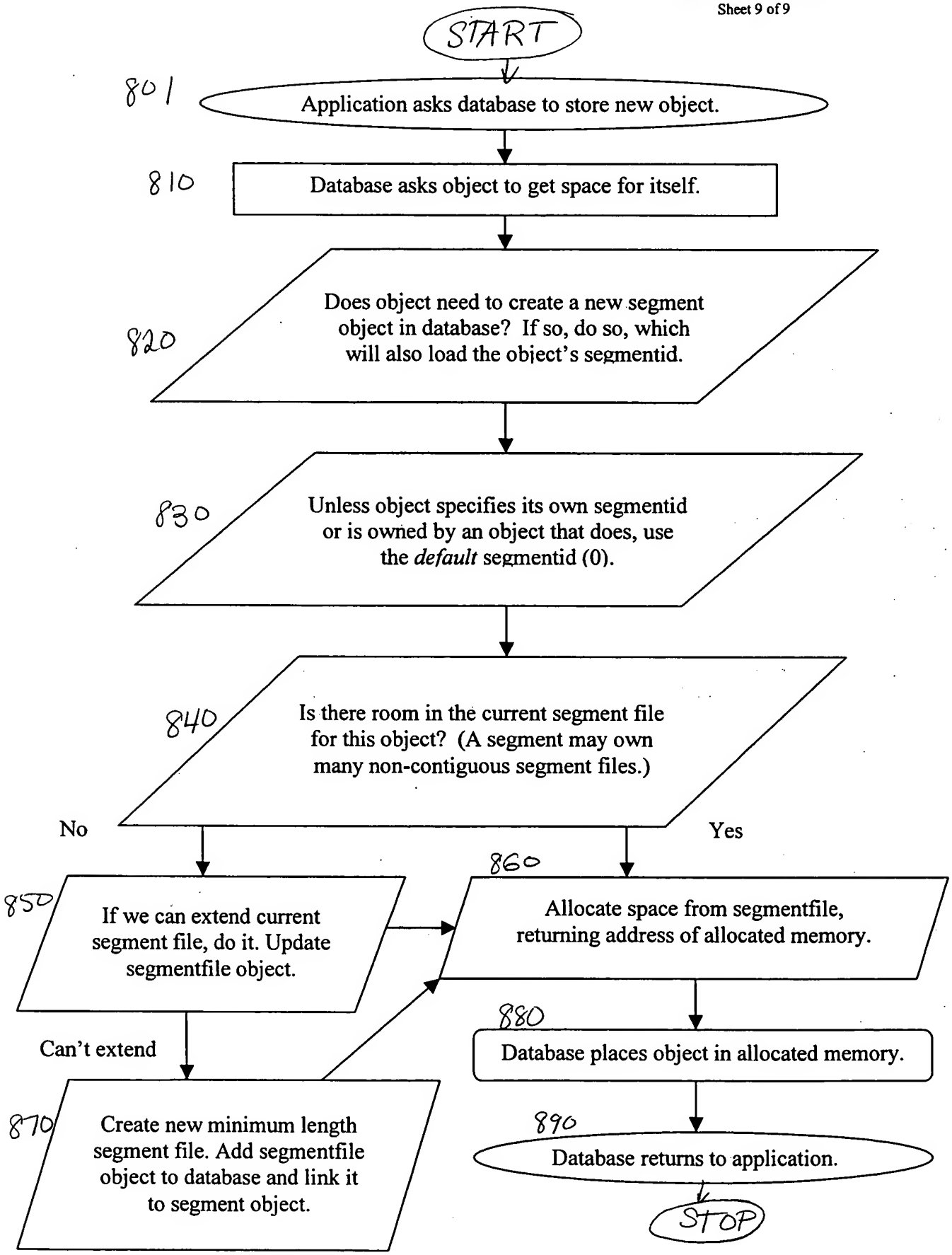

START
740
Application references an object in data repository.
750
Is the memory segment containing the object already mapped into memory?
Yes
No
755
Application continues normally.
STOP
760
Computer hardware issues segmentation fault.
765
Interrupt handler catches segmentation fault. Is segmentation fault address in space controlled by repository?
No
770
Return control to operating system or other fault handlers
STOP
Yes
775
If needed, enforce memory constraints by unmapping another segment.
780
Restart application at exact interrupt location.
Segment files are named in such a way that they can be located quickly based on the address of the object that segfaulted.
785
Find disk segment file representing address space segment. Map file.
739

FIG. 7B


START
801
Application asks database to store new object.
810
Database asks object to get space for itself.
820
Does object need to create a new segment object in database? If so, do so, which will also load the object's segmentid.
830
Unless object specifies its own segmentid or is owned by an object that does, use the *default* segmentid (0).
840
Is there room in the current segment file for this object? (A segment may own many non-contiguous segment files.)
No
Yes
850
If we can extend current segment file, do it. Update segmentfile object.
Can't extend
860
Allocate space from segmentfile, returning address of allocated memory.
870
Create new minimum length segment file. Add segmentfile object to database and link it to segment object.
880
Database places object in allocated memory.
890
Database returns to application.
STOP

800

FIG. 8